The Exchange Network: An Architecture for the Negotiation of Non-Repudiable Token Exchanges

Emanuel Palm, Olov Schelén, Ulf Bodin
Luleå University of Technology,
Luleå, Sweden
E-mail: {$firstname.lastname$}@ltu.se

Richard Hedman
Volvo Group Trucks Operations,
Göteborg, Sweden
E-mail: {$firstname.lastname$}@volvo.com

***Abstract*—Many use cases coming out of initiatives such as *Industry 4.0* and *Ubiquitous Computing* require that systems be able to cooperate by negotiating about and agreeing on the exchange of arbitrary values. While solutions able to facilitate such negotiation exist, they tend to either be domain-specific or lack mechanisms for non-repudiation, which make them unfit for the heterogeneity and scale of many compelling applications. In this paper, we present the *Exchange Network*, a general-purpose and implementation-independent architecture for digital negotiation and non-repudiable exchanges of *tokens*, which are symbolic representations of arbitrary values. We consider the implications of implementing the architecture in three different ways, using a common database, a blockchain, and our own *Signature Chain* data structure, which we also describe. We demonstrate the feasibility of the architecture by outlining our own implementation of it and also describe a supply-chain scenario inspired by one transportation process at Volvo Trucks.**

I. Introduction

With the realization of trends such as *Industry 4.0* [1] and *Ubiquitous Computing* [2], more computing devices are becoming interconnected than ever before. While the coming wave of smart machines may be able to facilitate a plethora of compelling use cases, we believe many of them will be economic in nature. Whether in smart manufacturing, value-chain integration, or product life-cycle analysis [1], goods, services, data, money, or other assets may have to change owners for a given use case to become viable. Every change of ownership is always preceded by some form of negotiation, whether it be accepting a delivery or bartering about a price, and the exchange may have to yield a receipt or other proof. While these negotiations could be handled by humans talking or writing to each other, as we assume to be typical nowadays, a digitized solution results in machines being able to monitor, assist or even participate in the negotiation process.

Systems facilitating negotiation and exchange do exist, with applications such as securities trading [3], resource access [4], and e-procurement [5]. However, these solutions make many assumptions about who is in control of the system, who must be trusted, and what can be negotiated about, which make them unfit for use cases outside of their intended application domains. Other recent efforts build on blockchains or other distributed ledgers to guarantee non-repudiation. However, most of these seem to either assume that a domain-specific negotiation protocol is enough [6] or require that code contracts be written for every automatable use case [7] [8] [9].

In this paper, we present the *Exchange Network* (EN), a general-purpose and implementation-independent architecture for the negotiation and exchange of token ownerships, facilitating a form of marketplace where both humans and computers can (1) negotiate, (2) exchange arbitrary assets or other commitments, and (3) prove that past exchanges have taken place. We show how different ways of implementing the architecture, which is defined in terms of abstract components and messages only, have diverging implications on governance, privacy, the credibility of proofs and system scalability. In particular, we briefly present an implementation example based on the *Signature Chain* data structure, a type of distributed ledger that facilitates privacy by not requiring peers to share records of their interactions with others. Furthermore, we compare the implementation to two other possibilities: one based on a common database and the other based on a permissioned blockchain system [10]. We also describe how an EN can facilitate a simple supply-chain use case in which a transport operations unit coordinates transports with a carrier.

A primary objective of our research efforts is to identify an *unobtrusive* architecture for digital cooperation. We assume that this architecture provides constructs with strong and well-understood analogies in real-world practices, which also do not diverge behaviorally from their real-world counterparts in significant ways. Cooperations can be transient or perpetual, remain unchanged for long periods of time or be renegotiated frequently, have strict privacy requirements or be carried out in public. Additionally, cooperation takes place in settings where different means of adjudication are available, making it relevant to ensure that the applied system is compatible with whatever means of litigation, arbitration, or peer judgement is available. To realize an architecture able to represent such characteristics, we chose *ownership* as the major construct and *negotiation* as the means of changing ownerships and then explore how that decision might affect the properties of any would-be implementations. Rather than building on an existing negotiation protocol, such as FIPA00037 [11], in which parties agree about *actions* to perform, we decided to design our protocol own around the concept of owned *tokens*, which are symbolic representations of arbitrary values. As we assume it is generally desirable to prove who owns what tokens, e.g., in courts of law, we also consider how different types of architecture implementations affect that ability.

II. The Exchange Network

An EN is either a monolithic or distributed application that facilitates a digital marketplace where well-known types of assets can be negotiated about, exchanged, and proven to have been part of past exchanges. Concretely, an EN facilitates coordinated changes to the owners of *tokens*, which could be thought of as symbolizing certain rights or obligations, such as the right of ownership, the obligation to render a service, or the obligation to pay. The architecture facilitates this process through four components, shown in Figure 1.

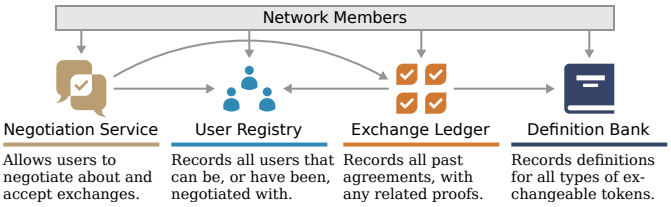

Fig. 1. Exchange Network components. Arrows denote usage.

Before presenting each of these components in turn, we would like to stress that we make no assumptions about how they store data or coordinate user interactions, as long as data can be accessed and users can interact. The components fulfill abstract functions that can be realized in multiple ways. Later in Section IV, we consider three ways of implementing the components and describe how each way has its own implications on governance and data distribution, as outlined in Figure 2, as well as interaction proofs and performance.

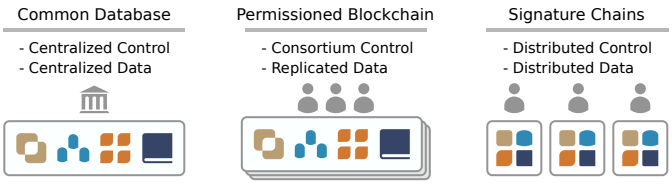

Fig. 2. The three considered ways to implement ENs. Note that replication and distribution of data are different in that the former requires each consortium member to own a more or less complete copy of all data, while the latter implies that data are shared only as needed. These implementations are further described in Section IV.

A. Negotiation Service

The first component we consider is the *Negotiation Service* (NS), which allows the users of an EN to propose, accept and reject exchanges. It relays *proposals* between pairs of negotiating users, which take turns trying to formulate a proposal that both deem acceptable.[1] If such an acceptable proposal can be identified by those users, the NS submits it to the *Exchange Ledger* (EL) component, which makes sure it can be proven to have taken place to any relevant third party, such as courts of law, insurance agencies, lenders, partners, and so on.

[1] We limit ourselves to negotiations between only two users to avoid making the procedure too complicated. While negotiations with more users may be quite relevant to many scenarios, we leave the topic for future research.

We describe the negotiation procedure in terms of three phases: (1) *qualification*, (2) *acceptance* and (3) *finalization*, which are depicted as a naive state machine in Figure 3.

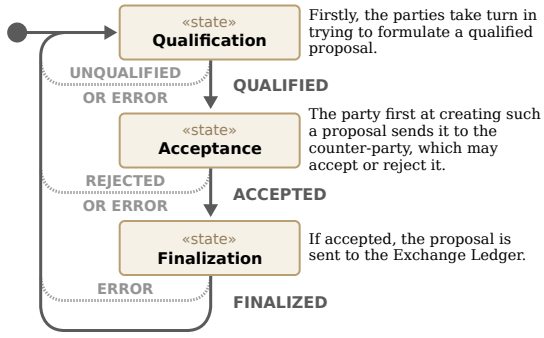

Fig. 3. A state machine showing how two negotiating users could progress from an initial proposal to an accepted and finalized one. A negotiation can be terminated at any time by either participant. Additionally, any number of negotiations can be ongoing at the same time between every pair of users.

1) Qualification: When a user has found another user that may provide one or more goods, services, or other assets of interest, the first objective is to find a *qualified* proposal believed to be acceptable to both. A qualified proposal is one that leaves no room for ambiguity regarding who would own what, should the proposal be accepted. The proposal is found by having the negotiating users take turn trying to formulate it. If not enough information is available for a candidate proposal to be qualified, an unqualified proposal may be used instead. Unqualified proposals may refer to abstract types of assets, include alternatives, or identify undesired assets. To facilitate the communication required to send these proposals, the NS provides the data types in Figure 4.

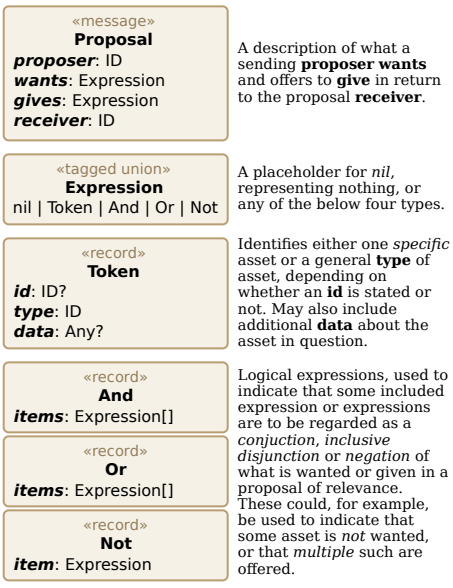

Fig. 4. The **Proposal** message and associated data types. **ID** represents an arbitrary identifier type, question marks (**?**) are used for optional values, while brackets ([]) are used to denote array types. Note that the types and fields represent a minimally viable set of proposals, not all useful such.

The possibility of using the **And**, **Or** and **Not** types, shown in Figure 4, allows users to formulate proposals analogous to those humans make while negotiating. Consider the example in Figure 5.

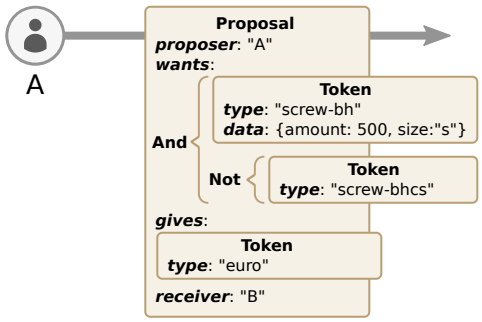

Fig. 5. An example of a *unqualified* proposal.

The example could be thought of as a digital version of the human request *"Can I have a package of 500 small button head screws? Make sure it is not the cap screw kind. I can pay in Euro."* A possible answer to this request is depicted in Figure 6, which could be transliterated as *"I could give you this package of 600 button head machine screws for €4.50."*

Fig. 6. Another *unqualified* proposal is sent as a reply to that in Figure 5.

Let us assume that the first user deems the counter-offer acceptable, carries only a 5 Euro bill, but does not mind giving away the change. That proposal is depicted in Figure 7 and is the first example to be qualified. Because it refers only to tokens with **id** fields and uses no **Or** or **Not** expressions, it is clear who would own what if the proposal would be accepted.

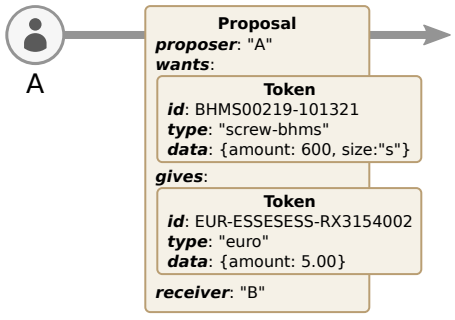

Fig. 7. A *qualified* proposal intended as a reply to the proposal in Figure 6.

Technically, a proposal is qualified if and only if it satisfies the **IsProposalQualified** function outlined in Listing 1.

```
function IsProposalQualified(proposal):
    return IsExpressionQualified(proposal.wants)
       and IsExpressionQualified(proposal.gives)

function IsExpressionQualified(expression):
    if expression is Token:
        return IsTokenQualified(expression)
    else if expression is And:
        foreach item in expression.items:
            if not IsExpressionQualified(item):
                return false
        return true
    else:
        return false

function IsTokenQualified(token):
    return token.id ≠ nil
```

Lst. 1. Functions for determining if a **Proposal** is *qualified*. A qualified proposal must contain only **Token** and **And** instances, and each **Token** must have an **id**. See figure 4 for type definitions.

Before we continue, we would like to stop and highlight how the proposed system of token expressions makes it possible to formulate logically impossible, or *unsatisfiable*, unqualified proposals, such as *"I want wrench B103, but I do not want wrench B103."* Systems dealing with arbitrary proposals may find it relevant to be able to detect unsatisfiable proposals using an SAT solver [12] or otherwise. Qualified proposals should, however, not be subject to this problem. Formulating unsatisfiable proposals requires one asset to be both wanted/given and not wanted/given at the same time, while qualified proposals do not allow the use of **Not** expressions.

2) Acceptance: As soon as one user formulates a qualified proposal, the objective becomes to determine if the counter-party also deems that proposal acceptable. While it may seem rather straightforward, it could require extra steps, depending on the NS implementation. Steps include providing signatures or first submitting the proposal to a special verifier service. After having sent a qualified proposal, the counter-party must either reject it by sending a new counter-proposal or accept it using the message in Figure 8.

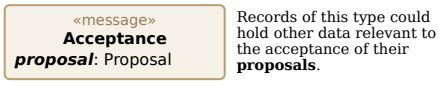

Fig. 8. Message used to accept a received qualified proposal. The proposal is rejected by sending a new counter-proposal or terminating the negotiation.

Another way to signal disinterest could be terminating the negotiation. If a counter-proposal is sent or received, the negotiation returns to the **Qualification** phase.

3) Finalization: When a qualified proposal has been formulated and accepted, it is submitted by the NS to the Exchange Ledger. The users are notified when it is known whether submission succeeded or failed, after which the negotiation returns to the **Qualification** phase. If there is more to negotiate about, negotiation continues. In any other case, the users are free to terminate the negotiation session.

B. User Registry

The second logical component, the *User Registry* (UR), is tasked with knowing (1) the *internal* identity and (2) the *external* identities of each EN user. If may provide individual users access to some or all of that information.

1) Internal Identity: The internal identity allows the EN to refer to a given user, which is what fundamentally enables the network to express that a particular asset belongs to a certain user. What type of internal identifiers are used will depend on the implementation the UR. If both the UR and the Exchange Ledger are hosted by a trusted authority, common integers would likely suffice. In the blockchain and SCs examples in Section IV, public keys [13] would have to be used.

2) External Identity: External identities, on the other hand, allow users to recognize other users outside the bounds of the EN. To determine where a user is physically located, it may be required to know where to deliver an exchanged good or to whom to render a service. Other details of relevance could be company identifiers, tax numbers, or contact details, which could become relevant in the case of a dispute, to assess user trustworthiness, or to contact a user using a different platform. How external identities are verified or whether multiple such identities are allowed per user depends on the UR implementation.

C. Exchange Ledger

The third logical component, the *Exchange Ledger* (EL), allows each user to (1) determine if proposed or already finalized ownership exchanges are *sound* and (2) prove that past ownership exchanges have taken place. While an EL could perhaps fulfill these responsibilities in multiple ways, we conceptualize it as doing so by maintaining and granting access to a history of **Exchanges**, as shown in Figure 9.

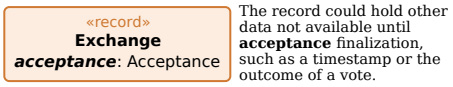

Fig. 9. Records an accepted and finalized ownership exchange. We also refer to these records as *agreements*.

1) Exchange Soundness: In particular, for a given ownership exchange to be sound, it must be known whether

1) the identities of the exchanging parties can be trusted,
2) the proposer owns the *given* tokens, unless created,
3) the acceptor owns the *wanted* tokens, unless created, and
4) the exchanged tokens are well-defined, and their regulations conformed to, as described in Section II-D.

While we are leaving room for an EL to reject unsound exchanges as part of negotiation finalization, as described in Section II-A3, it may or may not guarantee that all soundness properties are satisfied for each finalized exchange. It might, for example, be difficult to make guarantees about external regulations being adhered to, as explained later in Section II-D2. Users should always validate proposals of concern by themselves to limit the room for mistakes or other errors.

2) Exchange Proof: Courts of law, insurance agencies, partners, and other parties may be interested in seeing proof that particular ownership exchanges have taken place. How these proofs are facilitated by a particular EL depends on its implementation. The architecture makes no other assumption than that there is some way to present such proofs. We consider how these proofs could be facilitated in Section IV.

D. Definition Bank

The fourth and last logical component is the *Definition Bank* (DB). Its main task is to define the implications of owning or creating each type of token, especially in terms of what may be done with the token and the asset it represents. The DB component could be seen as a dictionary, allowing EN users to look up **Definitions**, as outlined in Figure 10, by their names, hashes, or other identifiers.

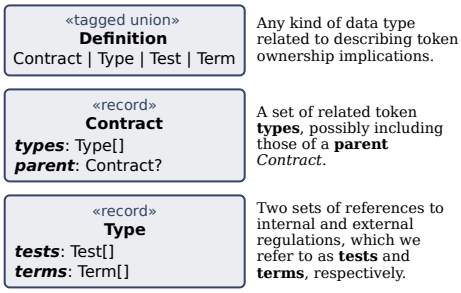

Fig. 10. The proposed types of DB definitions. Our naive **Contract** contains only **Types**, implying that it can verify contractual events but not facilitate them. This situation is in contrast to systems such as Ethereum [14], where contracts contain executable code.

1) Internal Regulation: These regulations, which we also refer to as *tests*, ensure that tokens cannot be abused inside an EN. A test could be thought of as a function taking a proposal, an EL and a DB as arguments, returning true only if the proposal is sound. For example, tests could limit the number of times a certain type of token can change owners, restrict creation or ownership of specific tokens to a fixed set of eligible users, or set expiration dates after which some tokens may no longer be exchanged. In other words, they could be used to prevent some unsound ownership exchanges from taking place at all.

2) External Regulation: These exist to ensure that the assets represented by any EN tokens are not abused outside the bounds of the EN. We refer to these regulations as *contractual terms* or just *terms*. For example, let us assume that two EN users have exchanged one token representing the right to a vehicle repair for another representing a promise of payment. At this point, there is no way for the EN itself to determine if any vehicle is repaired or any payment is made, as these events happen outside the EN's computers. This situation could be mitigated by ensuring that the types referenced by the exchanged tokens contain contractual terms honored by some legal authority, perhaps in the form of legal prose. As long as the exchange itself counts as proof, which we consider in Section IV, that authority could be used to resolve disputes.

III. Signature Chains

To demonstrate the viability of the EN architecture, we now present an implementation designed to maximize the opportunity for exchanges to be kept private. Concretely, the implementation is intended to mimic the way common paper contracts and other forms of signed instruments are used. Such instruments are typically known only to two agreeing parties until the event of a dispute, in which case the instruments are revealed to a legal authority or other arbitrator.

Our implementation operates without mediation, meaning that no set of parties needs to see and ratify each finalized exchange. It relies on a data structure we named the *Signature Chain* (SC), which uses cryptographic signatures and hashing [13] to ensure the (1) authorship, (2) order and (3) definitions of an exchange be denied or altered after its finalization.[2]

A. Implementation

Our system consists of a *node* both serving a web client and communicating with other nodes, as depicted in Figure 11.[3]

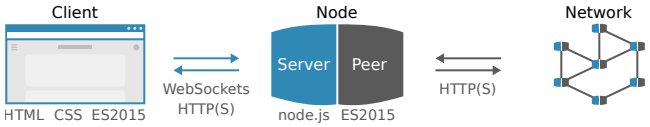

Fig. 11. The general design of our EN implementation. Humans operate each *Node* using a web browser *Client*. Every *Node* uses an internal *Server* to both provide its *Client* with static HTTP(S) [15] resources and send runtime data via WebSockets [16]. Each *Node* also contains a *Peer* module, which is used to communicate with the *Peer* modules of other *Nodes* over HTTP(S). The design requires no central data repository or any centralization of control.

Both the node and the client it serves are coded in TypeScript [17], which compiles to ES2015 (JS) [18] before execution. The JS of the node is executed by the node.js runtime [19], while the client HTML [20], CSS [21] and JS are executed by a web browser. We used these technologies and standards mostly because they are familiar to us. There are no inherent reasons why they should be technically superior to any other particular sets of technologies.

The client allows human users to manage negotiations; formulate, modify, accept and reject proposals; list finalized exchanges; list tokens together with the users that own them; and list the users themselves; among other things. It also performs proposal satisfiability tests, mentioned also in Section II-A1.

While the implementation indeed works and can demonstrate the SC concept, some important delimitations were made to reduce implementation effort. For example, all User Registry and Definition Bank data are provided at node startup and cannot change during runtime. Additionally, no communications are encrypted, and client users are not authenticated or authorized.

[2] The data structure has significant similarities to the *transaction* type employed by R3 Corda [9]. They can both refer to arbitrary definitions and previous interactions by hash, and may also be signed by two parties. Corda transactions, however, carry *state objects*, while SCs carry token exchanges.

[3] Available at https://github.com/emanuelpalm/en-signature-chains-poc. The paper describes GIT commit `694e3a73a1fbae67b9c106d47bd5`.

B. Data Structure

An SC is a chain of records, where each record *may* refer to (1) a previous related record and (2) a definition of relevance. Each record is cryptographically signed [13] by one or more *attestors*, in our case, a proposer and acceptor, and every reference to either a record or definition is the cryptographic hash of that data [13]. By implication, a third party given a chain of records with any associated definitions becomes able to verify that the records

1) indeed have been signed by their attestors,
2) were created in a certain order, and
3) always have referred to the provided definitions.

Rather than chains of records being stored in a centralized or replicated repository, each possible pair of EN users maintains and extends its own sets of chains, as depicted in Figure 12. This procedure leaves room for each pair of users to maintain privacy, given that they can agree on not sharing their mutual records with others. Thus, both users of each pair can independently reveal any shared chain to any party of interest, such as a court of law, a partner or another party.

Each possible pair of users may keep their own sets of signature chains, denoted here by edges A, B and C.

Fig. 12. The distinct sets of SCs of a three-user EN.

To concretely implement the data structure, a given EN may need to make the messages in Section II able to form chains of signatures, which could be realized by amending the **Proposal** and **Acceptance** types as described in Figure 13.

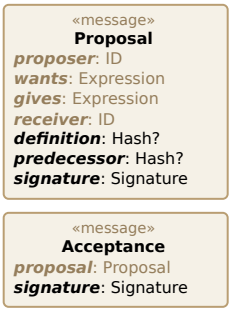

Fig. 13. Amended variants of messages first described in Figures 4 and 8.

Additionally, each relevant EN definition type, such as the ones in Figure 10, ought to refer to its subdefinitions via their hashes. An example of such an SC is illustrated in Figure 14.

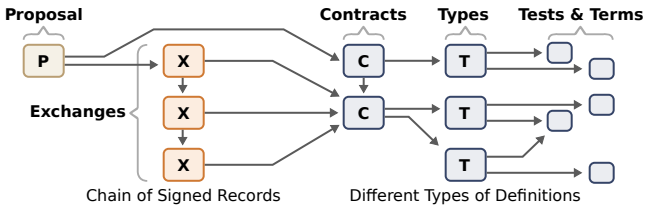

Fig. 14. An example EN SC. Arrows denote references by hash.

IV. Comparisons of Possible Implementations

The Exchange Network (EN) architecture leaves room for many kinds of concrete implementations. Here, we consider how three possible implementations would affect (1) governance, (2) privacy and data distribution, (3) proofs of interactions and (4) system scalability. The implementation from Section III is included in the comparison. We summarize the properties of the implementations in Table I.

TABLE I
Properties of considered possible EN implementations.

	Common Database	*Blockchain*	*Signature Chains*
Governance	Trustee	Consortium	None
Data Distribution	Centralized	Replicated	Distributed
*Interaction Proof**	Trustee Word	Vote	Signatures Only
Scalability	Bound-by-Database	Bound-by-Vote	Unbounded

*Cryptographic signatures can always be used, as in Section III-B. Note that in blockchain systems, transactions are typically signed only by their issuers by default, as in [22] and [23]. A satisfactory proof requires the signatures of both agreeing parties.

A. Common Database

Building an EN around a traditional database, such as MySQL [24], could allow strict control of the members of a given network, as well as rigorous soundness checks of all finalized exchanges. If negotiations are relayed through a centralized Negotiation Service, ongoing negotiations could also be monitored and verified. In either case, a single party must be entrusted with maintaining the network. Due to its appointment and position, the trustee's word could count as proof of interaction, if considered trustworthy. At least the acceptor, proposer and trustee must know of each finalized exchange. System scalability would be limited primarily by the underlying database.

B. Permissioned Blockchain

If instead using a system such as Hyperledger Fabric [22] as a foundation, a consortium rather than a single party is entrusted with maintaining the system while enabling the same strict member control and soundness checks as above. However, this process comes at the cost of having to replicate and vote on all data, which significantly limits system scalability. It also means that each consortium member must know of each finalized exchange. While there may be ways to limit interaction visibility through cryptography, those ways would likely also limit the opportunity for exchange soundness to be verified by the consortium. If a fact of significance has been seen and ratified by each maintaining member, a majority testimony could be used as proof that the event has occurred.

C. Signature Chains

Without any centralized control, network members are themselves responsible for determining what other parties can be trusted and for ensuring exchange soundness. As no trustee or consortium can testify to the exchange's credibility, only cryptographic signatures can be used as proof. There is no inherent limit to scalability, as there is no global synchronization point. Only the acceptor and proposer of each exchange must know that it has occurred.

V. Supply Chain Use Case

To make the utility of the EN architecture more apparent, we present here an example use case involving the transportation of truck components, akin to how Volvo Trucks currently manages such transports. Some EN implementation is used to propose, accept and register each completed interaction between some *Carrier* (**C**) and a *Transport Operations* (**TO**) unit. The carrier takes a number of components from a manufacturer to some assembly plant, as directed by the **TO**. The purpose of each EN negotiation is to establish a new set of rights and obligations as a result of some contractual term being fulfilled, using the EN messages we describe in Sections II-A1 and III-B. The four interactions proceed as follows.

1) *Call-Off*: **TO** sends an *EN Propose* to **C**, wanting **C** to accept the obligation to transport a given number of components, guarantee a particular delivery time, and insure the components while in transit. If **C** believes the requested transport capacity will be unavailable, it may reject the proposal or propose another delivery time. We assume **C** replies with *EN Accept*.
2) *Transport Request*: **TO** sends one *EN Propose* to **C** for each individual component, requesting cross-docking and transportation. Each message specifies a pick-up time, a serial number, and sequencing information, which are used to ensure that components are delivered in the order of assembly. **C** would normally commit to each request via an *EN Accept* but could reject them in case of complications. Such early rejections would allow **TO** to immediately search for alternatives to avoid costly delays at the assembly plant.
3) *Pick-Up*: **C** then sends one *EN Propose* to **TO** at the time of loading and departure of each individual component, which would normally be accepted via *EN Accept* messages. Rejections could indicate mismatches in tracking data, perhaps due to human errors. Automatically detecting such errors could save the time and costs that would normally be incurred by manual inspection.
4) *Delivery*: Upon arrival to the assembly plant, **C** sends and *EN Propose* to **TO**, wanting the delivery to be confirmed, which it does by sending an *EN Accept* only if the conditions agreed upon in steps 1 and 2 are met. If, for example, a component would be out of sequence, **TO** could make a counter-proposal for the carrier to agree on a new deal for the deviating item. This setup could lead to faster deviation agreements, easier follow-ups and reduced costs.

While details regarding the obligations of **TO** to **C**, this use case should illustrate how an EN could be used to automatically handle possible deviations online and without human intervention. As an EN is used, all completed negotiations are registered on a shared *Exchange Ledger*, which we hope can be used as evidence in the case of a dispute. Additionally, **C** could use finalized exchanges not yet paid for as a guarantee of future income, which, for example, could be useful when negotiating interest rates with a bank.

VI. Discussion

The concepts we present in this paper could be a significant step towards a paradigm in which machines are increasingly able to monitor, assist and autonomously participate in the economy. To make the remaining steps of that journey more apparent, we discuss here (A) shortcomings of our design, (B) the idea of trustless systems, and (C) how our architecture could fit into the context of industry.

A. Design Shortcomings

1) Ambiguity of Ownership: What does it really mean to be the proven owner of a digital token? This problem is fundamental not only to EN tokens but also to paper contracts and other signed instruments. For example, what is a deed of ownership really worth? The answer is that it depends on whether the token or instrument in question is *honored*. This honor is typically established by ensuring that litigation, or some other form of adjudication, is possible in the case of a dispute. Because each party knows that any counter-party can take legal action, a tangible incentive exists to obey the terms of any agreement. For our architecture to be practically useful, effort needs to be spent on formalizing the token tests and terms of Section II-D both to avoid the risk of parties interpreting tokens differently and to ensure that courts of law or other adjudicators can be used if desired.

2) Limits to Negotiation Expressiveness: For an EN to be able to replace human-to-human negotiation, its Negotiation Service (NS) must be able to represent any expression a human could utter in such a context. While we demonstrated how it could represent three expressions in Section II-A, we know of cases that cannot be easily represented. For example, *"I want at least 1000 small screws and will pay no more than €20."* One way to approach the issue could be to use a corpus of human negotiations and then extend the NS specification until all negotiations in that corpus can be represented.

3) Involving Secondary Authorities: Having access to one or more trusted authorities can be critical for collaboration to become practically possible. Courts of law, private arbitration firms, insurance agencies, money lenders, or inspection firms could be of relevance to establish trust between parties. In Section IV-B, we showed how more than one authority could be part of maintaining the same EN but believe it will be difficult to gather all useful authorities in the same consortium. A more realistic approach could be to extend the architecture to allow the involvement of secondary authorities, which could veto or approve proposals during negotiation finalization.

4) Dynamic Definition Creation: In Section I, we implied that smart contract systems such as [7], [8] and [9] require that code-as-contracts are installed *before* they can be used. In contrast, the abstract negotiation protocol we present in Section II does not require definitions to be in place before collaborations can start. As every interaction is a negotiation and the result of every negotiation can be regarded as a new definition, negotiations may result in the creation of new contracts, amendments or exceptions. However, we do not explore how this process can be facilitated in this paper.

5) Regulating User Identities: We made no assumptions in this paper about how party identification should be regulated while knowing it is an important and delicate issue. Future work should consider how to prevent identity abuse, which could lead to real-world entities being able to avoid being held accountable for their actions.

B. Trustless is Not Enough

Readers from the blockchain community may react to our seeming ignorance of blockchain systems being *trustless* [25], which implies they remove the need to rely on trusted middlemen. While blockchain systems indeed do this, they only do it to an extent. Claiming that a system is trustless is the same as saying that it relies on a network of voting computers instead of a traditional authority. Such computer networks are currently unable to perform all useful functions that traditional authorities can. In particular, contemporary blockchain systems are (1) largely limited to acting on signed facts that they cannot verify beyond system boundaries and (2) wield no other fundamental power than deciding what can be recorded in their ledgers. In Bitcoin [23], this arrangement is sufficient to maintain account balances and prevent incorrect transactions, but it is not enough to punish fraudulent users for fooling others into sending them money. In contrast, traditional authorities can make rational decisions regarding the truthfulness of facts, and they could compel misbehaving users into conformance by invoking the power of a police force. Consequently, we do not see how the current state-of-the-art in blockchain technology would be sufficient for typical industrial use cases without also involving trusted authorities.

C. Industry Integration

While our solution may help enable entirely new economic use cases, we deem it most relevant to first consider how existing economic processes can be digitized. Digital negotiation and ownership exchange could lead to benefits that are generally in line with process digitization, such as reducing the time needed to complete contractual interactions with new or existing partners or being able to track and analyze those interactions in real-time. Such improvements could lead to (1) increased room for asset accountability, (2) more fine-grained economic forecasting and (3) reduced capital requirements.

There are, however, some roadblocks that need to be cleared before industry adoption can begin. We have already mentioned compatibility with legal authorities and arbitrators, as well as with insurance agencies, money lenders, and inspection firms, which are just a few examples of all potentially relevant authorities. Compatibility with these parties will likely require considerable legal effort both to make the technology lawful and to establish collaboration models and best practices for different industries. Another major roadblock is finding a suitable EN implementation. We have already mentioned that we believe that this implementation needs to be as non-intrusive as possible on existing business models and practices. We had this objective in mind when we designed the SC implementation in Section III, but it is far from complete.

VII. Conclusions

The EN architecture we proposed in this paper constitutes a generic model for asset transfer, where each asset is represented by a unique digital *Token*. We have shown that a possible implementation of the model based on the SC data structure could ensure a high degree of privacy between peers and facilitate horizontal scalability, allowing it to meet the performance requirements and heterogeneity of industrial applications and global markets.

The primary objectives of this work are (1) unobtrusiveness, (2) implementation independence, and (3) reusability. The first objective we address by building on the ideas of negotiation and ownership to formulate the EN domain model. The latter two we approach by separating our architecture into four abstract components, which can be described as follows.

- *Generic Asset Negotiation and Transfer*: The NS component provides a clearly defined model for collaboratively refining offers into concrete and tentative asset transfers, which can ultimately be executed atomically and logged immutably. In essence, this conceptual model provides an open high-level protocol specification for *asset-for-asset* transfer negotiations, where an asset represents any form of right or obligation. Like any protocol, it could be part of many kinds of applications and be supported by many different protocol implementations.
- *User Identity Tracking*: The UR component keeps track of other EN members in terms of both their internal and external identities. The former allows users to be identified within an EN system, while the latter helps anchor EN members to legal entities or other forms of identities outside the same EN.
- *Exchange Regulation*: The DB component stores definitions, serving to programmatically and legally define the implications and regulations associated with each kind of exchangeable asset. It fulfills this role by storing regulations, which are both used to verify exchanges and serve as proof of any violations.
- *Exchange Record-Keeping*: Finally, the EL component is tasked with storing an immutable history of ownership exchange records while guaranteeing that each meets the constraints and requirements imposed by the other components. Each record in this ledger serves as proof that a described interaction has taken place and could be useful as evidence if provided to a third party.

It is our belief that the architecture in this paper, or a solution like it, could prove pivotal for digitizing economic interactions between industries and within society at large.

Acknowledgments

We would like to thank Caroline Berg von Linde, Johan Hörmark, Christian Lagerkvist and Jamie Walters at SEB for their indispensable insights regarding banking, money and financial services. We would also like to thank Richard Verbeet at the University of Ulm for making us aware of FIPA00037. This work was funded via the *Productive 4.0* project (EU ARTEMIS JU grant agreement number 737459).

References

[1] L. D. Xu *et al.*, "Industry 4.0: state of the art and future trends," *International Journal of Production Research*, vol. 56, no. 8, 2018. [Online]. Available: https://doi.org/10.1080/00207543.2018.1444806

[2] M. N. O. Sadiku, Y. Wang, S. Cui, and S. M. Musa, "Ubiquitous computing: A primer," *International Journal of Advances in Scientific Research and Engineering*, vol. 4, no. 2, 2018. [Online]. Available: https://doi.org/10.7324/IJASRE.2018.32611

[3] V. F. Minton, "Interactive securities trading system," U.S. Patent 6 014 643, January 11, 2000.

[4] R. Gavriloaie, W. Nejdl, D. Olmedilla, K. E. Seamons, and M. Winslett, "No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web," in *European Semantic Web Symposium*. Springer, 2004, pp. 342–356. [Online]. Available: https://doi.org/10.1007/978-3-540-25956-5_24

[5] P. H. Ketikidis, A. Kontogeorgis, G. Stalidis, and K. Kaggelides, "Applying e-procurement system in the healthcare: the EPOS paradigm," *International Journal of Systems Science*, vol. 41, no. 3, pp. 281–299, 2010. [Online]. Available: https://doi.org/10.1080/00207720903326878

[6] J. J. Sikorski, J. Haughton, and M. Kraft, "Blockchain technology in the chemical industry: Machine-to-machine electricity market," *Applied Energy*, vol. 195, pp. 234–246, 2017. [Online]. Available: https://doi.org/10.1016/j.apenergy.2017.03.039

[7] A. Norta, "Self-aware smart contracts with legal relevance," in *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018. [Online]. Available: https://doi.org/10.1109/IJCNN.2018.8489235

[8] Y. Zhang and J. Wen, "The IoT electric business model: Using blockchain technology for the internet of things," *Peer-to-Peer Networking and Applications*, vol. 10, no. 4, pp. 983–994, July 2017. [Online]. Available: https://doi.org/10.1007/s12083-016-0456-1

[9] M. Hearn. (2016) Corda: A distributed ledger. Accessed 2019-02-22. [Online]. Available: https://www.corda.net/content/corda-platform-whitepaper.pdf

[10] M. E. Peck, "Blockchains: How they work and why they'll change the world," *IEEE Spectrum*, vol. 54, no. 10, pp. 26–35, 2017.

[11] "FIPA communicative act library specification," Foundation for Intelligent Physical Agents, FIPA 00037, 2002, accessed 2019-05-09. [Online]. Available: http://www.fipa.org/specs/fipa00037

[12] N. Eén and N. Sörensson. The minisat page. Accessed 2019-01-24. [Online]. Available: http://minisat.se

[13] A. Salomaa, *Public-key cryptography*, 2nd ed., ser. Texts in Theoretical Computer Science. Springer Science & Business Media, 1996.

[14] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014, accessed 2019-02-01. [Online]. Available: http://gavwood.com/paper.pdf

[15] R. Fielding *et al.*, "Hypertext transfer protocol (HTTP/1.1): Message syntax and routing," Internet Request for Comments, RFC Editor, RFC 7230, 2014. [Online]. Available: http://rfc-editor.org/rfc/rfc7230.txt

[16] I. Fette and A. Melnikov, "The WebSocket protocol," Internet Request for Comments, RFC Editor, RFC 6455, December 2011.

[17] G. Bierman, M. Abadi *et al.*, "Understanding TypeScript," in *European Conference on Object-Oriented Programming*. Springer, 2014.

[18] A. Wirfs-Brock, "ECMAScript 2015 language specification," ECMA International, ECMA-262, 2015. [Online]. Available: http://www.ecma-international.org/ecma-262/6.0

[19] S. Tilkov *et al.*, "Node.js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, 2010.

[20] S. Faulkner *et al.*, "HTML 5.2," W3C, Working Draft, 2017. [Online]. Available: https://www.w3.org/TR/2018/WD-html53-20180809

[21] T. Atkins and S. Sapin, "CSS syntax module level 3," W3C, W3C Candidate Recommendation, February 2014. [Online]. Available: http://www.w3.org/TR/2014/CR-css-syntax-3-20140220

[22] E. Androulaki, A. Barger *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, pp. 30:1–30:15. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[23] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, accessed 2019-02-08. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[24] C. Bell, *Introducing the MySQL 8 Document Store*. Apress, 2018. [Online]. Available: https://doi.org/10.1007/978-1-4842-2725-1

[25] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016. [Online]. Available: https://doi.org/10.1109/ACCESS.2016.2566339